CLAIMS

I claim:

1. A method, comprising:

forming a dictionary containing base entries representing individual instructions in a program and sequence entries representing corresponding sequences of multiple instructions in the program; and

generating items that represent the program in terms of the base entries and the sequence entries.

2. A method as recited in claim 1, wherein the forming comprises creating a split-stream dictionary.

3. A method as recited in claim 1, wherein the sequence entries represent short sequences consisting of two to four instructions.

4. A method as recited in claim 1, wherein the sequence entries represent sequences of multiple instructions that are used multiple times in the program.

5. A method as recited in claim 1, wherein the generating comprises:

comparing an input string of instructions to the sequence entries in the dictionary; and

if the input string matches a particular sequence entry, generating an item that references the particular sequence entry.

6. A method as recited in claim 1, wherein the generating comprises:

comparing progressively smaller strings of multiple instructions, where each string begins with a first instruction, to the sequence entries in the dictionary;

if any string of multiple instructions matches a particular sequence entry, generating a first item that references the particular sequence entry; and

if no string of multiple instructions matches the sequence entries, generating a second item that references a base entry associated with the first instruction.

7. A method as recited in claim 1, further comprising compressing the dictionary.

8. A method as recited in claim 1, further comprising compressing the base entries of the dictionary.

9. A method as recited in claim 8, wherein the compressing comprises:

sorting the base entries by opcodes to create instruction groups so that there is one instruction group for each opcode; and

for each instruction group, sorting the base entries according to size of individual instruction fields and outputting each instruction field as a separate stream.

10. A method as recited in claim 1, further comprising compressing the sequence entries of the dictionary.

11. A method as recited in claim 10, wherein the compressing comprises constructing tree structures for individual sequences of multiple instructions.

12. A computer readable medium storing the dictionary and the items produced as a result of the method as recited in claim 1.

13. A computer readable medium having computer-executable instructions that, when executed on one or more processors, performs the method as recited in claim 1.

14. A method, comprising:

analyzing a program containing multiple instructions;

creating base entries in a dictionary for individual instructions; and

creating sequence entries in the dictionary for corresponding sequences of multiple instructions that are used multiple times in the program.

15. A method as recited in claim 14, wherein the sequence entries represent short sequences consisting of two to four instructions.

16. A method as recited in claim 14, further comprising compressing the base entries of the dictionary.

17. A method as recited in claim 16, wherein the compressing comprises:

sorting the base entries by opcodes to create instruction groups so that there is one instruction group for each opcode; and

for each instruction group, sorting the base entries according to size of individual instruction fields and outputting each instruction field as a separate stream.

18. A method as recited in claim 14, further comprising compressing the sequence entries of the dictionary.

19. A method as recited in claim 18, wherein the compressing comprises constructing tree structures for individual sequences of multiple instructions.

20. A method as recited in claim 14, further comprising generating items that represent the program in terms of the base entries and the sequence entries.

21. A method as recited in claim 20, wherein the generating comprises:

comparing progressively smaller strings of multiple instructions, where each string begins with a first instruction, to the sequence entries in the dictionary;

if any string of multiple instructions matches a particular sequence entry, generating a first item that references the particular sequence entry; and

if no string of multiple instructions matches the sequence entries, generating a second item that references a base entry associated with the first instruction.

22. A computer readable medium storing the dictionary produced as a result of the method as recited in claim 14.

23. A computer readable medium having computer-executable instructions that, when executed on one or more processors, performs the method as recited in claim 14.

24. A method, comprising:

creating base entries in a dictionary for individual instructions in a program;

creating sequence entries in the dictionary for corresponding sequences of multiple instructions that are used multiple times in the program;

compressing the base entries and the sequence entries to produce a compressed dictionary; and

generating items that represent the program in terms of the base entries and the sequence entries.

25. A method as recited in claim 24, wherein the sequence entries represent short sequences consisting of two to four instructions.

26. A method as recited in claim 24, wherein the compressing comprises:

sorting the base entries by opcodes to create instruction groups so that there is one instruction group for each opcode; and

for each instruction group, sorting the base entries according to size of individual instruction fields and outputting each instruction field as a separate stream.

27. A method as recited in claim 24, wherein the compressing comprises constructing tree structures for individual sequences of multiple instructions.

28. A method as recited in claim 24, wherein the generating comprises:

comparing progressively smaller strings of multiple instructions, where each string begins with a first instruction, to the sequence entries in the dictionary;

if any string of multiple instructions matches a particular sequence entry, generating a first item that references the particular sequence entry; and

if no string of multiple instructions matches the sequence entries, generating a second item that references a base entry associated with the first instruction.

29. A method as recited in claim 24, further comprising decompressing the compressed dictionary.

30. A method as recited in claim 29, further comprising translating the items back to the instructions by using the base entries and the sequence entries of the dictionary.

31. A computer readable medium having computer-executable instructions that, when executed on one or more processors, performs the method as recited in claim 24.

32. A method for decoding a file derived from a program, the file having a dictionary with base entries representing individual instructions in the program and sequence entries representing corresponding sequences of multiple instructions in the program and multiple items that represent the program in terms of the base entries and the sequence entries, the method comprising:

recovering the base entries and the sequence entries of the dictionary; and

translating the items to instructions in the program by using the base entries and the sequence entries in the dictionary.

33. A method as recited in claim 32, wherein the dictionary is compressed and the recovering comprises decompressing the compressed dictionary.

34. A method as recited in claim 32, wherein the translating comprises copying the base entries and the sequence entries into a code buffer.

35. A computer readable medium having computer-executable instructions that, when executed on one or more processors, performs the method as recited in claim 32.

36. A computer readable medium having computer-executable instructions that, when executed on one or more processors, directs a computing device to:

read a program containing multiple instructions;

create base entries in a dictionary for individual instructions in the program;

create sequence entries in the dictionary for corresponding sequences of multiple instructions that are used multiple times in the program; and

generate items that represent the program in terms of the base entries and the sequence entries

37. A computer readable medium as recited in claim 36, wherein the sequence entries represent short sequences consisting of two to four instructions.

38. A computer readable medium as recited in claim 36, further comprising instructions to compress the dictionary.

39. A computer readable medium as recited in claim 36, further comprising instructions to:

sort the base entries by opcodes to create instruction groups so that there is one instruction group for each opcode; and

for each instruction group, sort the base entries according to size of individual instruction fields and outputting each instruction field as a separate stream.

40. A computer readable medium as recited in claim 36, further comprising instructions to compress the sequence entries by constructing tree structures for individual sequences of multiple instructions.

41. A computer readable medium as recited in claim 36, further comprising instructions to:

compare progressively smaller strings of multiple instructions, where each string begins with a first instruction, to the sequence entries in the dictionary;

if any string of multiple instructions matches a particular sequence entry, generate a first item that references the particular sequence entry; and

if no string of multiple instructions matches the sequence entries, generate a second item that references a base entry associated with the first instruction.

42. A program compression architecture comprising:

a dictionary builder to construct a dictionary containing base entries representing individual instructions in a program and sequence entries

representing corresponding sequences of multiple instructions that are used multiple times in the program; and

an item generator to generate items that represent the program in terms of the base entries and the sequence entries.

43. A program compression architecture as recited in claim 42 wherein the sequence entries represent short sequences consisting of two to four instructions.

44. A program compression architecture as recited in claim 42 wherein the item generator is configured to compare an input string of instructions to the sequence entries in the dictionary and if the input string matches a particular sequence entry, generate an item that references the particular sequence entry.

45. A program compression architecture as recited in claim 42 wherein the item generator is configured to compare progressively smaller strings of multiple instructions, where each string begins with a first instruction, to the sequence entries in the dictionary such that (1) if any string of multiple instructions matches a particular sequence entry, the item generator produces a first item that references the particular sequence entry and (2) if no string of multiple instructions matches the sequence entries, the item generator produces a second item that references a base entry associated with the first instruction.

46. A program compression architecture as recited in claim 42 further comprising a dictionary compressor to compress the dictionary.

47. A program compression architecture as recited in claim 46 wherein the dictionary compressor is configured to compress the base entries independently of the sequence entries.

48. A program compression architecture as recited in claim 46 wherein the dictionary compressor is configured to sort the base entries by opcodes to create instruction groups so that there is one instruction group for each opcode, the dictionary compressor being further configured to sort the base entries within each instruction group according to size of individual instruction fields and output each instruction field as a separate stream.

49. A program compression architecture as recited in claim 46 wherein the dictionary compressor is configured to construct tree structures for individual sequences of multiple instructions.

50. An embedded system comprising the program compression architecture of claim 42.

51. A computer comprising:

a memory;

a processing unit coupled to the memory; and

a program compression system stored in the memory and executable on the processing unit, the program compression system building a dictionary containing base entries representing individual instructions in a program and sequence entries representing corresponding sequences of multiple instructions in the program, the program compression system generating items that represent the program in terms of the base entries and the sequence entries.

52. A computer as recited in claim 51, wherein the program compression system is further configured to compress the dictionary.

53. A data structure stored on a computer readable medium, comprising:

base entries representing individual instructions in a program; and

sequence entries representing corresponding sequences of multiple instructions that are used multiple times in the program, the sequence entries referencing the base entries.

54. A data structure stored as recited in claim 53, further comprising items that reference the base entries and the sequence entries to represent instruction strings in the program.